HALMA - Dokumentacja Projektu

Antoni Przybylik
PIPR 22Z
GRUPA 101

21 stycznia 2023

Zadanie

Napisać program grający w Halma. Powinna być możliwość gry:

- dwóch osób ze sobą,
- osoby z komputerem,
- komputera z komputerem.

Program powinien kontrolować poprawność wykonywanych ruchów. Interfejs z użytkownikiem może być tekstowy.

Bardzo istotną częścią zadania jest opracowanie i zaimplementowanie jak najlepszego algorytmu grającego w grę.

Moduł realizujący algorytm gry komputera musi być wydzielony.

Cel i opis projektu

Celem projektu było zaprojektowanie i napisanie gry Halma w języku Python, przygotowanie testów jednostkowych i dokumentacji. W tym celu należało wykorzystać umiejętności zdobyte na przedmiocie „Podstawy Informatyki i Programowania”.

Architektura

Projekt jest podzielone na trzy główne moduły:

- Silnik gry (klasa Engine).
- Interfejsy użytkownika.
- Klasy graczy (klasy pochodne od Player).

W silniku zaimplementowano bazową funkcjonalność gry. UI odpowiada za komunikację z użytkownikiem. W ramach projektu zaimplementowałem tylko jedno UI (w formie TUI), ale aplikacja jest tak zaprojektowana, że istnieje możliwość stworzenia wielu UI, które będą komunikować się z resztą aplikacji w taki sam sposób jak HalmaTui (klasa implementująca interfejs TUI). Klasy graczy, natomiast, odpowiadają za wykonywanie ruchów, mogą to być boty lub ludzie. Kiedy użytkownik zarząda wykonania ruchu wciskając odpowiedni przycisk w UI wywoływana jest metoda make_move() dla obecnie ruszającego się gracza. Jeśli ten gracz jest botem - po prostu wykonuje ruch. Gracz-Człowiek ma możliwość wyświetlenia okna dialogowego z zapytaniem o ruch.

W aplikacji wydzieliłem też trzy moduły pomocnicze:

- Silnik UI.
- Interfejs silnika gry dla użytkownika.
- Klasę Game reprezentującą stan całej gry.

Oddzielenie Silnika UI (w klasie TuiEngine) od modułu wyświetlającego UI ma przyczynę w tym, że elementy UI mają mieć możliwość wyświetlać także niektóre obiekty klas pochodnych od Player (gracze-ludzie).

Interfejs silnika gry dla użytkownika (w klasie GameInterface) ma za zadanie tworzyć warstwę abstrakcji pomiędzy UI, a silnikiem gry. Tu przekazywany jest napis, który wpisał użytkownik w okienku dialogowym i rozbijany jest on na konkretne współrzędne i wykonywany jest ruch wykorzystując niżej-poziomowy interfejs silnika gry (w klasie Engine).

Klasa Engine implementuje bazową funkcjonalność gry. Znajdują tam się między innymi metody odpowedzialne za ustawianie/odczyt pól na planszy, metody pozwalające na sprawdzenie możliwych ruchów z danego pola. W tej klasie jest zapisany stan planszy i numer obecnego ruchu.

Klasa HalmaTui implementuje tekstowy interfejs użytkownika (TUI) dla gry Halma. Używa on ogólnych elementów interfejsu takich jak okna dialogowe, które są dostarczane przez silnik TUI (w klasie TuiEngine).

Klasy pochodne od Player implementują metodę make_move(), która odpowiada za wykonywanie ruchu. W przypadku botów ruch jest po prostu wykonywany. Gracze-ludzie wyświetlają okno dialogowe przy użyciu interfejsu klasu TuiEngine (tej samej której używa HalmaTui) pytającego gracza o ruch, a potem go wykonują.

Klasa GameBot dziedzicząca po Player jest klasą bazową dla wszystkich botów (RandomBot, ForwardBot). Zawiera metody wspólne dla wszystkich botów. W klasach pochodnych od GameBot są zaimplementowane konkretne algorytmy dla każdego bota.

Klasa TuiPlayer dziedzicząca po Player implementuje wykonywanie ruchu przez człowieka. Wyświetla okno dialogowe z zapyteniem o ruch, wprowadzony ruch przekazuje do modułu GameInterface.

Klasa TuiEngine zawiera metody takie jak: wyświetlająca okno dialogowe, wyświetlająca splash screen. Jest wykorzystywana przez HalmaTui do narysowania okna gry i przez TuiPlayer do zapytań o ruch.

Klasa GameInterface wprowadza warstwę abstrakcji pomiędzy UI, a silnik gry. Na przykład ruchy wpisane przez użytkownika nie są sprawdzane i przetwarzane na współrzędne w tych samych funkcjach które wyświetlają UI tylko są przekazywane klasie GameInterface, która ma referencję na obiekt klasy Engine i wykonuje ona ruch. Referencję na obiekt klasy GameInterface ma UI i TuiPlayer, który jest obiektem klasy Player implementującym grę przez użytkownika.

Klasa Game posiada referencję na obiekty klas: GameInterface, Engine i obiekty reprezentujące gracza białego i czarnego (obiekty klas pochodnych od Player). Ten stan można zapisać do pliku i wczytać go z pliku.

Wymagania

Żeby móc uruchomić grę w trbie TUI należy mieć zainstalowaną bibliotekę curses.

Interfejs TUI działa tylko w terminalach obsługujących 8-bitowe kolory i pozwalających na zmianę ich wartości[1]. Dodatkowo, jest wymagane żeby okno terminala miało rozmiar co najmniej 40x70[2].

Instrukcja użycia

W ramach projektu jest zaimplementowany interfejs tekstowy (TUI). Grę w trybie TUI otwieramy uruchamiając plik wykonywalny halma-tui.py.

Komunikacja z programem odbywa się przez okna dialogowe. Są ich dwa rodzaje: Pytające o wybór opcji z listy i pytające o inny ciąg znaków.

W oknie dialogowym pierwszego rodzaju należy wprowadzić w polu tekstowym numer wybranej opcji i zatwierdzić klawiszem enter.

Inaczej postępujemy z oknem dialogowym pytającym nas o ruch. W nim należy wprowadzić ruch w formacie „AB-CD”[3], gdzie AB to współrzędne pola z krórego chcemy wykonać ruch, CD to współrzędne pola na które chcemy wykonać ruch. Współrzędne to para literek którymi podpisane są pola z boku planszy - pierwsza współrzędna z lewej strony planszy, druga współrzędna na górze.

RYSUNEK TOPOLOGIA

[1] Każdy nowoczesny terminal jak urxvt, xfce4-terminal, gnome-terminal powinien spełniać te wymagania. Gra nie działa w xtermie.

[2] Wysokość x Szerokość.

[3] Wielkość liter nie ma znaczenia, nie należy natomiast wprowadzać spacji wewnątrz zapisu.

Część refleksyjna

Aplikacja ma przejrzystą architekturę z wyraźnie oddzielonymi modułami. W zależnościach modułów nie ma cykli.

Interfejs użytkownika jest skalowalny i dostosowywuje się do rozmiaru terminala. Sterowanie jest intuicyjne, a kolory są dobrane precyzyjnie z dbałością o przejrzystość okna gry.

Gra ma możliwość zapisu i wczytania z pliku. Są zaimplementowane dwa boty - losowy i sprytny. Są duże możliwości rozszerzenia funkcjonalności dzięki podziałowi na moduły. Przykładowo, można zaimplementować GUI które komunikowało by się z resztą aplikacji przez taki sam interfejs jak już napisany tekstowy interfejs. Można zaimplementować więcej botów, a nawet możliwość gry zdalnej.

Jest nawet możliwość doimplementowania innej wersji TUI na bazie interfejsu klasy TuiEngine.